Filtragem Colaborativa de Correio Electrónico não Solicitado

Artur Machado[1], Paulo Cortez[2], Pedro Sousa[3]

[1,3]*Departamento de Informática/CCTC*
Universidade do Minho, Braga, Portugal
email: ajpcm1@gmail.com, pns@di.uminho.pt

[2]*Departamento de Sistemas de Informação/ALGORITMI*
Universidade do Minho, Guimarães, Portugal
email: pcortez@dsi.uminho.pt

***Resumo* — O serviço de correio electrónico assume uma crescente importância na sociedade actual e, em particular, no âmbito dos serviços de comunicação que hoje em dia são alvo preferencial de utilização por parte dos utilizadores da Internet. No entanto, são ainda muitos os desafios que se colocam a este serviço, nomeadamente no que diz respeito à crescente proliferação do fenómeno de correio electrónico não solicitado. Desta forma, este trabalho posiciona-se no desenvolvimento de um sistema colaborativo, o qual permite aos utilizadores finais contribuir para a obtenção de mecanismos de detecção de *spam* mais eficientes. Isto será realizado através do desenvolvimento de um sistema que possibilite, de uma forma simples e flexível, o envio de filtros de *spam* dos utilizadores finais para servidores especializados, de modo a tornar possível o acesso e a troca colaborativa de filtros entre os diferentes utilizadores de rede. Esses filtros serão posteriormente combinados, de diferentes formas, nos clientes dos utilizadores finais, com vista a uma melhoria dos processos de classificação e detecção de *spam*.**

***Palavras-chave*:** Correio Electrónico, *Spam*, Filtros Colaborativos.

I. INTRODUÇÃO

A crescente proliferação de correio electrónico não solicitado (*spam*) é um problema grave que afecta actualmente as redes de comunicações. Este problema tem inúmeras razões, entre as quais uma forte motivação económica, que advém da possibilidade de aceder a um elevado número de consumidores a um baixo custo [1,2]. A título exemplificativo alguns estudos apontam para que nos dias de hoje aproximadamente 50% a 80% de todo o tráfego mundial de *e-mail* seja *spam* [3,4]. Assim, este problema atinge os indivíduos e as organizações, devido à invasão de privacidade, à divulgação de burlas e vírus informáticos [5] e ao aumento do tráfego na rede e ao tempo gasto a ler mensagens indesejadas [1,6]. Para minorar este problema existem diferentes abordagens, como as colaborativas (e.g. *blacklists*) e as baseadas em processos de filtragem (análise de conteúdo de e-mail via técnicas de *text mining*) [7].

Assim, é neste contexto que se posiciona este trabalho, e os seus contributos estão associados à definição e ao desenvolvimento de uma arquitectura distribuída que permita auxiliar no desenvolvimento de estratégias colaborativas de filtragem de correio electrónico. No que se refere à estrutura básica do sistema a desenvolver, este deverá basear-se em extensões a clientes do serviço de correio electrónico usualmente utilizados na Internet. Estas extensões, que deverão ser fáceis de instalar e configurar, permitirão o desenvolvimento de funcionalidades adicionais nos clientes de correio electrónico através de uma fácil interface com o utilizador [8]. Estas extensões deverão então permitir a interacção com servidores específicos, quer para submissão de amostras de *spam* recebido, quer para troca de filtros de *e-mail*. A disponibilização e organização dos filtros nos servidores poderão ter em conta a classificação dos diferentes participantes em diversos grupos de interesse por forma a se constituírem como comunidades e redes sociais de indivíduos que, quer pessoal quer profissionalmente, partilham interesses e objectivos comuns. Desta forma, a partir do sistema projectado neste trabalho, será então possível desenvolver diferentes estratégias colaborativas de detecção e filtragem de *spam* e estudar a eficácia das mesmas.

II. A PROBLEMATICA DO SPAM

A. O que é o Spam e como se distribui?

Um utilizador que use o *e-mail* com uma elevada frequência geralmente recebe dois grupos distintos de mensagens: as que são solicitadas (*ham*) e as que não são solicitadas (*spam*) como já foi referido anteriormente. As mensagens solicitadas são aquelas que o utilizador espera receber, ou até mesmo não esperando receber, são mensagens que o utilizador quererá ler, tendo pois um grau significativo de importância para ele. Por oposição, as mensagens chamadas *spam* ,ou não solicitadas, são mensagens de publicidade contendo por vezes vírus, *worms,* ou com algum prejuízo para o sistema, sendo distribuídas em massa e sobre as quais o utilizador não possui controlo relacionado com a sua recepção. O *spam,* geralmente enviado e controlado por indivíduos denominados por *spammers*, tem como principais objectivos:

- A divulgação para venda de produtos por vezes ilícitos, como medicamentos (de prescrição obrigatória) ou drogas;
- Os actos de *phishing,* i.e. fraude electrónica com o intuito de se obter informações pessoais tais como *passwords* ou números de cartões de crédito, fazendo-se passar por empresas de confiança;
- A intrusão no sistema do utilizador, para que seja roubada informação deste ou, por vezes, usar o sistema como porta para distribuição de mais *spam*;
- Destruição do sistema do utilizador por via de vírus.

Estas intenções na distribuição do *spam* atingem não só os utilizadores (como alvo primário), mas também os *ISPs* por serem as entidades responsáveis por controlar os servidores

SMTPs, os quais vão realizar a entrega das mensagens. Por um lado, os utilizadores desperdiçam demasiado tempo e recursos a transferir e verificar essas mensagens, são contaminados e enganados pelo conteúdo delas e perdem espaço de armazenamento no servidor de *e-mail*, que muitas vezes é limitado. Por sua vez, os *ISPs* também são afectados. Estes têm que dispensar espaço de armazenamento e largura de banda, com custos associados, para que haja a capacidade para as mensagens serem guardadas e, sendo grande parte destas mensagens do tipo *spam*, estes custos são desnecessários.

Para entender a gravidade deste problema, na actualidade o *spam* corresponde a sensivelmente 75%-80% de todo o tráfego de mensagens de *e-mail* [4]. Estes valores reflectem-se sempre no utilizador, visto ser ele o destinatário final. Em média, um utilizador passou de uma recepção diária na ordem dos 3.7 mensagens de *spam*, em 2001, para 6.2 por dia [9]. Obviamente, estes números estão associados a custos para o utilizador que por ano na recepção de *spam* atingem os 30 euros [10]. Custos estes que resultam da necessidade de maior largura de banda para a transferência das mensagens, e espaço de armazenamento tanto no cliente bem como no servidor. São várias as técnicas utilizadas pelos *spammers* para enviar *spam*, nas quais se podem destacar as seguintes [11]:

- *Spam* Directo: Utilização, pelos *spammers*, de *ISPs* que não têm políticas de controlo de *spam*, ou que não aplicam nenhuma penalização pela sua prática;
- *Open Relays:* São servidores de *e-mail* (e.g. *SMTP*) em que não é necessária autenticação para enviar mensagens. Normalmente são explorados para o envio de *spam* visto que os mesmos mantêm o anonimato e são de fácil acesso;
- *Botnets:* Este é o método preferido dos *spammers*. As *Botnets* são um conjunto de máquinas controladas por um nó central. Esta máquina central, geralmente controlada pelo *spammer,* usa as restantes máquinas para o envio das mensagens. Normalmente, os nós "filhos" são criados por intermédio de *worms* que se auto-propagam e permitem que o *spammer* a use como *mail relay*. Um exemplo deste *worm* é o "*Bobax"*. Este auto propaga-se por *e-mail* para outras futuras vítimas. Outros *botnets* muito utilizados pelos *spammers* são o *"Agobot"* e o *"SDBot"* [12].

Para combater este problema existe um investimento crescente por parte dos *ISPs* e dos utilizadores mais especializados na área da informática. Ambas as partes esforçam-se para encontrar formas de tornar ineficazes todos os métodos utilizados pelos *spammers* assim como desencorajar a sua prática. Os *ISPs* tentam também encontrar uma maneira de tornar o *spam* mais dispendioso, mas enquanto isso não vai sendo possível investem na filtragem do *spam* mesmo antes deste estar disponível para o utilizador. Por parte do utilizador, apenas há a necessidade de se esforçar para filtrar o *spam* que chega do servidor, para não perder tempo a ver *e-mails* que não quer, ou até mesmo para evitar ser infectado por vírus.

B. Técnicas Actualmente Existentes de Combate ao Spam

As técnicas *anti-spam* são inúmeras e encontram-se em permanente mudança, face às alterações das técnicas dos *spammers*. Existem soluções de diferente natureza como o caso da aplicação de processos judiciais em alguns países para penalizar quem envia *spam*, das tentativas de aumentar o custo do *spam* para que se torne inviável o seu envio e do recurso a filtros *anti-spam*, já citados anteriormente.

Em alguns países já se começam a implementar leis para penalizar criminalmente a prática de *spam*. Por exemplo, em Portugal já é considerado crime desde 2004, visando a proibição do uso de *spam* para propostas de marketing directo, excepto se o destinatário o consentir. Retrocedendo no tempo, se formos até Julho de 2002, observa-se que foi aprovada uma directiva de Privacidade e de Comunicação Electrónica pela União Europeia [4] na qual a lei Portuguesa se baseia.

Uma outra técnica seria impor custos acrescidos ao *spam*. Estes custos poderiam ser baseados, por exemplo, em pagamentos para ler e enviar *e-mails* com quantias simbólicas em dinheiro "virtual", aumentando assim o custo do envio e fazendo com que seja menos atractivo o envio de *spam* [10].

Estas duas técnicas anteriormente referidas apresentam falhas e são difíceis de colocar em prática. Como consequência, surgem então os filtros como ferramenta alternativa. Estes facilitam a distinção entre o que é *spam* e o que é *ham* através da análise do conteúdo do *e-mail* (métodos heurísticos) e pelos testes realizados a listas ou repositórios globais de *spam* (métodos colaborativos).

Os métodos colaborativos, normalmente, dependem de uma rede de utilizadores que criam em conjunto listas, sob a forma de repositórios, nas quais constam vários endereços/domínios ou assinaturas de mensagens identificadas como sendo *spam* [3, 4, 11, 13] (e.g. as *blacklists* globais). Quando um utilizador recebe uma nova mensagem é calculada a assinatura (em forma de *hash*) para que a privacidade seja assegurada. Esta assinatura é então enviada para um servidor e este procede à sua comparação com todas as existentes na sua lista. Caso a assinatura da mensagem se encontre no repositório será enviado ao utilizador um aviso de *spam* para essa mesma mensagem [14]. O grande problema que se encontra subjacente a este método é a enorme susceptibilidade de alterações das mensagens de *spam* [3]. Como as assinaturas das mensagens dependem dos caracteres, é suficiente a mudança de um destes para gerar uma assinatura diferente. Por tal facto, os *spammers* inserem caracteres aleatórios nas mensagens enviadas para que a sua assinatura seja alterada, o que faz com que esta não conste da lista e assim seja identificada como correio *ham*, o que gera um falso negativo. Em contrapartida, dado o facto deste método utilizar uma rede global de utilizadores, é apenas necessário que um utilizador identifique uma mensagem como *spam* para que toda a restante rede de utilizadores beneficie dessa *tag*, sendo as próximas mensagens com assinaturas compatíveis identificadas automaticamente. Actualmente existem disponíveis para *download* ferramentas que permitem a implementação deste método colaborativo. Um exemplo é o "*Spamato*" que oferece varias soluções dependentes do sistema operativo e do cliente de *e-mail* do utilizador possui.

Os filtros que possuem como base os métodos heurísticos são, regra geral, locais e localizam determinadas características das mensagens que permitem distingui-las, i.e., este método assume a possibilidade de encontrar diferenças entre as mensagens *ham* e *spam*. Existem vários métodos heurísticos baseados em diferentes características, como os

baseados na origem da mensagem, os baseados na análise do tráfico e os baseados no conteúdo da mensagem [10].

A classificação baseada na heurística da origem da mensagem ocorre sempre que o cliente de *e-mail* do utilizador começa a transferir a mensagem, usando o *header* para tal. Um exemplo que recorre a este tipo de heurística são as chamadas *blacklists* locais. No processo de transferência de uma mensagem é-lhe retirada o campo "*FROM:*" comparando-o com cada elemento de uma determinada lista. No caso de o endereço constar dessa lista, a mensagem fica automaticamente assinalada como *spam*, como referido atrás. Ao utilizador compete adicionar endereços de *e-mail* das mensagens que são *spam* à medida que as mensagens conseguem ultrapassar a barreira estabelecida pelo filtro, i.e. o utilizador é o responsável pela manutenção dessa mesma lista.

O filtro baseado na heurística da análise de tráfego é mais orientado para o *ISP* do que para o utilizador. Nesta técnica, através da análise de registos de tráfego *SMTP* identifica-se, quando a mensagem chega ao servidor ou é enviada, se o emissor da mensagem está a enviar em difusão para se poder classificar essa mesma mensagem [10].

Por último, a classificação baseada no conteúdo da mensagem ocorre após a recepção total do *e-mail*. A classificação, realizada através da utilização de técnicas *data mining*, passa pela identificação das características contidas na mensagem, tais como palavras, imagens, anexos ou algum outro conteúdo presente que possibilite aumentar as suspeitas. Deste modo, o filtro é capaz de compreender se o *e-mail* é *spam* marcando-o como tal. Para que seja possível a análise e a identificação, o filtro vai ter que aprender a distinguir essas características. Isto é realizado pela correcção, por parte do utilizador, de mensagens que foram classificadas como *ham* ou *spam* erradamente. Quando isto acontece, o filtro procede ao reconhecimento das características dessa mensagem bem como à sua assimilação, para que no futuro as mensagens que contenham características semelhantes, sejam imediatamente identificadas e classificadas correctamente. Os filtros mais populares utilizam geralmente o algoritmo *Naive Bayes*[1] (NB). Quando se emprega este algoritmo, o utilizador terá que treinar o filtro com *ham* e *spam,* até que este consiga distinguir correcta e autonomamente as mensagens que recebe. Normalmente, esta aprendizagem é efectuada através do número de ocorrências de cada palavra da mensagem, definindo assim o conjunto de características da mensagem. Na chegada de uma mensagem o filtro executa o algoritmo e classifica-a através da avaliação de algumas das suas palavras, resultando deste processo uma probabilidade da mensagem ser considerada *spam* [4,10,13, 15].

III. ARQUITECTURA DO SISTEMA

O sistema desenvolvido para o combate a esta problemática é de seguida apresentado, sendo composto por dois componentes distintos. O primeiro componente é constituído por acções de filtragem, com uma classificação que se baseia no conteúdo das mensagens. No entanto, aqui implementaram-se algumas alterações significativas no que diz respeito ao funcionamento normal de um filtro. Desta forma, na solução desenvolvida, em vez de um só filtro, recorre-se ao uso de vários filtros diferentes, e que foram treinados de maneira diferente e por vários utilizadores. Assim, a plataforma que foi desenvolvida para um cliente deste sistema terá um filtro baseado em conteúdo (que é treinado pelo próprio utilizador), ou *bag of words,* e um conjunto adicional de filtros pertencentes a outros utilizadores colaborativos que, sob anonimato, os disponibilizaram para o grupo em questão. Este primeiro componente ocupar-se-á também de elaborar estratégias inteligentes de combinação dos diferentes filtros partilhados e colocados ao dispor do utilizador final.

A segunda componente do sistema desenvolvido ocupa-se exactamente com a distribuição destes mesmos filtros para um servidor que, de forma anónima, selecciona quais os ficheiros a distribuir e para quem serão enviados, controlando também as políticas de acessos dos utilizadores. Estas duas componentes formam, no seu todo, uma abordagem baseada em métodos heurísticos, mas no entanto colaborativa, constituindo pois uma abordagem híbrida inovadora. De seguida serão explicados, pormenorizadamente, cada um dos componentes desenvolvidos, com apoio ilustrativo.

A. Arquitectura da Filtragem

Nos dias correntes, o uso dos filtros para a classificação do *spam* está a aumentar exponencialmente. Isto tem-se verificado por inúmeras razões, tais como pelo facto da possibilidade do utilizador treinar o seu próprio filtro bem como pelo facto dos algoritmos implementados apresentarem uma eficácia bastante boa, comparativamente a outras técnicas anteriormente apresentadas. Esta técnica apresenta contudo algumas falhas que impossibilitam uma filtragem completamente fidedigna. Estas falhas começam por ser notoriamente visíveis quando o filtro é criado a partir de um número reduzido de casos de treino. Nesta situação o filtro irá classificar erradamente o *spam* e o *ham*, resultando assim em diversos falsos negativos ou positivos. Esta situação também é verificada quando o utilizador treina mal o filtro. Um outro problema resulta das rápidas alterações das técnicas de *spam* utilizadas, em que o filtro, por não estar treinado para essas técnicas, obtém uma resposta errada. Por estas razões, as técnicas que se baseiam em filtros requerem, por parte do utilizador, um treino constante e eficiente para que estes não classifiquem erradamente as mensagens.

A solução aqui apresentada assume como parte integrante esta técnica de filtragem, mas de uma forma colaborativa, contribuindo para que os problemas apresentados possam ser minimizados. Neste contexto, este componente da arquitectura foi desenvolvido em extensões passíveis de serem integradas no cliente *Mozilla Thunderbird.* Adicionalmente, como base de trabalho foi utilizado um filtro já implementado numa extensão para esse cliente, e que se denomina por *Thunderbayes*[2], que utiliza o algoritmo *Naive Bayes* na classificação das mensagens. Este já apresenta uma filtragem típica que, a partir das propriedades da mensagem, compara-as ao ficheiro de treino e devolve ao cliente de *email* um resultado em percentagem reflectindo se a mensagem é ou não *spam*. O ficheiro de treino é composto por palavras (ou outras

[1] Em homenagem a *Thomas Bayes,* matemático Britânico do Séc. XVIII. O algoritmo foi implementado pela primeira vez num filtro de *email* em 1996.

[2] Disponível em https://addons.mozilla.org/en-US/thunderbird/addon/4025

características) pertencentes às mensagens que o utilizador classificou previamente como *ham* ou *spam*. A particularidade do *Thunderbayes* consiste na execução do algoritmo de *Naive Bayes* sob a forma de um *proxy*, fazendo com que o cliente se ligue a essa proxy, em vez da tradicional ligação directa ao servidor de email. Este *proxy*, além da classificação, também fará a ligação ao servidor de email, sendo então possível o envio e a recepção das mensagens.

Para que os problemas identificados anteriormente fossem minorados, alterou-se substancialmente este *proxy* de maneira a que fosse possível aceitar e usar não só o filtro estatístico do utilizador, mas também os vários filtros provenientes de outros utilizadores, resultando assim numa arquitectura nova. Esta particularidade apresentada confere ao cliente uma nova abordagem no treino. Desta forma, e visto que se utiliza a cooperação com outros utilizadores, é possível aumentar a eficácia da filtragem mesmo que o filtro individual do utilizador não esteja previamente treinado. De igual forma, mesmo no caso do filtro individual estar mal treinado ou aquando da recepção de um novo tipo de mensagem *spam*, será de esperar que esta estratégia consiga uma redução de falsos negativos ou positivos.

Na Figura 1, está representada a arquitectura aqui exposta, ilustrando cada passo tanto no treino como na classificação da mensagem no envio ou recepção. Nesta solução, o utilizador continuará a poder classificar manualmente as suas mensagens como *ham* ou como *spam*, dependo do que este considera como tal, bem como proceder a correcções de classificações erróneas. O cliente, segundo essa classificação, vai enviar, usando o protocolo *HTTP*, para a proxy, o resultado da classificação (0 se for *ham* ou 1 caso contrario) e as características da mensagem, para que o filtro aprenda as preferências do utilizador, tal como é indicado na Figura 1 como *passo 1*, construindo assim o seu próprio ficheiro de treino. No que se refere especificamente aos filtros, o utilizador tem disponível as opções de partilha do seu próprio ficheiro de treino, bem como a importação dos ficheiros de treino de outros utilizadores alocados num servidor remoto (este procedimento será explicado mais à frente). A construção e a importação de filtros constitui assim o *passo 2* ilustrado na

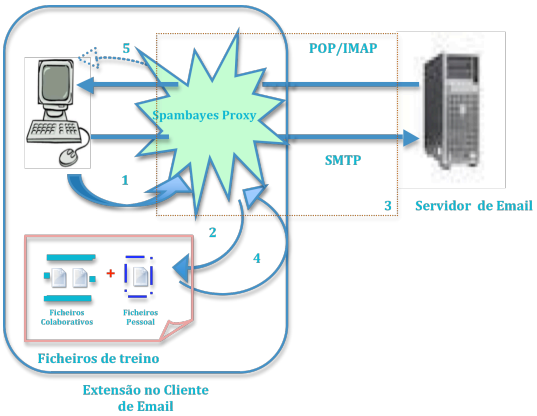

Fig. 1. Arquitectura desenvolvida para a classificação das mensagens

Figura 1. Assim, o filtro do cliente do utilizador vai ter então à sua disposição tanto o ficheiro pessoal de treino como os ficheiros colaborativos de outros utilizadores, podendo todos contribuir para o processo de filtragem.

Assim sendo, após a recepção de uma mensagem, por POP ou IMAP, o proxy verificará as características desta, (*passo 3* da Figura 1), classificando-a e enviando essa mesma mensagem para o cliente. A classificação ocorre quando as características da mensagem são comparadas com os ficheiros de treino (*passo 4*), devolvendo-se uma probabilidade da mensagem ser *spam*. Esta probabilidade é calculada através do uso do algoritmo *Naive Bayes*. Visto que são utilizados vários ficheiros, o algoritmo é executado para cada ficheiro obtendo-se assim diferentes probabilidades. Para então calcular a probabilidade total, o sistema desenvolvido recorre a um dos seguintes métodos, M1 e M2:

- *M1 – Media aritmética*: o resultado final é calculado através da media aritmética de todas as probabilidades;
- *M2 – Coeficiente de correlação de Pearson*: para cada ficheiro de treino importado, é calculado um peso (coeficiente) através da correlação de *Pearson* entre a probabilidade dada pelo filtro e a certeza dada pelo utilizador em relação a um conjunto de mensagens. Para que o processo seja automático, serão utilizadas as mensagens que foram consideradas no treino do filtro visto que será certo que, para o utilizador, estas são de certeza 0 ou 1 (*ham* ou *spam*). Assim, quando uma mensagem é recebida e a probabilidade de cada ficheiro for calculada, esta é multiplicada pelo coeficiente do ficheiro de treino, tal como demonstrado pela seguinte equação:

$$Rf(M) = \frac{\sum_i^n \alpha \Pr_i(M)}{n}, \quad (1)$$

em que *Rf* é o resultado final, *M* a mensagem, *n* o número total de ficheiros de treino, α o coeficiente da correlação e $\Pr_i(M)$ a probabilidade dada pelo filtro estatístico *i*.

Assim, por cada um destes métodos, o resultado final será calculado e enviado para o cliente em conjunto com a mensagem (*passo 5* da Figura1). Como o *proxy* é local, o envio desse resultado será simples, não envolvendo nenhum protocolo especifico, sendo pois uma simples actualização da base de dados do cliente, onde não só se guarda a mensagem mas também informação relacionada com a mensagem, tal como esta classificação. A extensão desenvolvida irá então ler o campo da base de dados correspondente e informar o utilizador sobre o resultado da classificação da mensagem.

B. *Arquitectura da Partilha de Filtros*

Para que a arquitectura da filtragem anteriormente proposta seja possível, será necessário possibilitar a partilha dos ficheiros de treino entre os diversos utilizadores, recorrendo-se neste caso ao seu armazenamento num servidor central. Este servidor poderá estar organizado de diferentes formas. Em particular, uma solução interessante será o agrupamento dos diversos filtros de acordo com os perfis e interesses dos diversos colaboradores, tirando também assim partido da crescente organização em redes sociais dos intervenientes na rede Internet. De seguida será descrita tecnicamente a arquitectura implementada para este componente da arquitectura.

Na Figura 2 está referida a arquitectura desenvolvida para a partilha de ficheiros de treino. Aqui são incluídos dois servidores distintos, o gestor de repositório, desenvolvido em *Java* e usando *sockets* para estabelecer a ligação com o cliente,

e o servidor de ficheiros no qual tem implementado os protocolos *Secure File Transfer Protocol (SFTP)* ou *File Transfer Protocol (FTP)*. Estes dois servidores, que podem ser localizados na mesma máquina ou em máquinas distintas, formam o repositório dos ficheiros e gerem as transferências dos diferentes ficheiros partilhados.

Inicialmente o cliente autentifica-se junto do gestor do repositório com os seus dados e informa o servidor da acção pretendida. Esta acção poderá ser *download* ou *upload*, se este quiser transferir a lista de ficheiros de treino do servidor para o cliente, ou se quiser transferir o seu próprio ficheiro para o servidor, respectivamente (correspondendo ao *passo 1* da Figura 2). Esta transacção é efectuada através do recurso à linguagem XML. De seguida o gestor irá verificar se o utilizador está correctamente identificado, e em caso afirmativo, formula uma resposta mediante o pretendido pelo utilizador. Se o utilizador pretender fazer o *download*, o gestor irá consultar a base de dados e construir uma listagem de todos os ficheiros disponíveis para o utilizador. No caso de o utilizador pretender fazer o *upload* do próprio ficheiro então o gestor consulta a base de dados para verificar o nome do ficheiro a armazenar. É importante nesta fase referir que, para preservar a privacidade, cada utilizador terá associado um nome de ficheiro aleatório, no qual não haverá nenhuma correspondência entre o utilizador e esse mesmo nome. Esta ligação à base de dados para verificação e formulação de listagens de ficheiros corresponde ao *passo 2* da Figura 2.

Após a formulação da listagem dos ficheiros a enviar ou a receber, o gestor constrói um ficheiro XML para ser enviado de novo ao cliente, sendo este representado pelo *passo 3*. Esta resposta conterá não só os nomes dos ficheiros a transferir, mas também o endereço do servidor do sistema de ficheiros, os dados de autentificação do servidor e o tipo de protocolo que este utiliza. Com estes dados, o cliente liga-se então ao servidor indicado pelo gestor utilizando a autentificação fornecida e indica que ficheiro é que pretende transferir nesse momento (*passo 4*). O servidor verifica a autentificação e se contém o(s) ficheiro(s) pretendido(s), iniciando a transferência em caso afirmativo, tal como indica no *passo 5* da imagem. Estes dois passos são repetidos para todos os elementos da

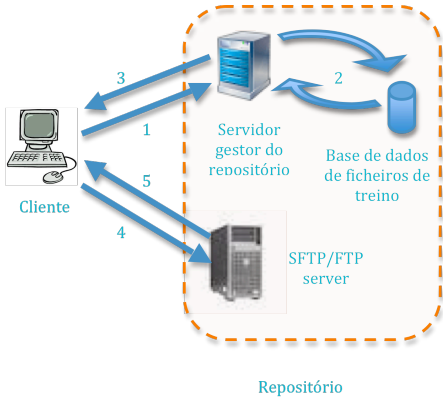

Fig. 2. Arquitectura desenvolvida para a partilha de ficheiros de treino

lista de ficheiros a transferir, tal como indicado pelo gestor do repositório. Após a transferência, um ficheiro de texto é armazenado no cliente, indicando quais os ficheiros de treino disponíveis neste. Desta forma, os ficheiros de treino ficam disponíveis na plataforma do cliente podendo contribuir para a classificação das mensagens que venham a ser recebidas.

A titulo meramente ilustrativo a Figura 3 apresenta algumas das alterações que foram introduzidas pela extensão desenvolvida no interface com o utilizador do *Mozilla Thunderbird*. Os comandos adicionados permitem proceder às operações de partilha e acesso a filtros que estão armazenados no servidor do serviço.

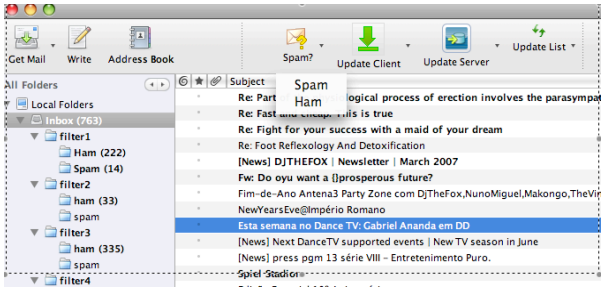

Fig. 3. Exemplos de comandos adicionados à interface do *Thunderbird* para acesso ao repositório de filtros (e.g. *update client*, *update server*, *update list*)

IV. Discussão de Resultados

Antes de se colocar o sistema desenvolvido sob utilização real, procedeu-se a testes laboratoriais específicos incidindo sobre questões relacionadas com as estratégias de filtragem. De seguida serão apresentados resultados de alguns dos testes preliminares que foram realizados, com o objectivo de recolher uma indicação inicial sobre a viabilidade do sistema proposto. Estes testes deverão ser alargados, no futuro, a outros repositórios públicos e a outros cenários de utilização, incluindo cenários específicos em que se considere a agregação de filtros pertencentes utilizadores com perfis e interesses semelhantes, o que se acredita que possibilitará ainda um melhor desempenho da solução proposta.

Os resultados aqui apresentados foram obtidos utilizando o repositório público *Enron* [16], usando mensagens de *ham* e *spam* de 5 utilizadores diferentes (Kam, Far, Bec, Lok e Kit) do *Enron*. Os testes foram efectuados utilizando duas métricas diferentes:

- AUC;
- TPR@FPR, com uma taxa de t=0.01.

A primeira métrica corresponde à área sob a curva (*Area Under the Curve –AUC)* da *Receiver Operating Characteristic – ROC*, sendo calculada com a seguinte expressão:

$$AUC = \int_0^1 ROC dD, \tag{2}$$

sendo D ∈ [0,1] um ponto de decisão, i.e., se a probabilidade for superior a D, então a mensagem é classificada como *spam*, caso contrário é *ham*. Quando maior for o valor de AUC, melhor é a capacidade de discriminação do modelo estatístico. Um classificador aleatório tem um AUC=0.5, enquanto que o modelo perfeito corresponde a um AUC de 1.0.

A segunda métrica apresenta a taxa de verdadeiros positivos (TPR) para uma dada taxa fixa de falsos positivos (FPR). Uma vez que na detecção de spam, os falsos positivos tem um custo superior, utiliza-se normalmente valores baixos de FPR. Nas experiências efectuadas, optou-se por um limite de erro de

TABELA I
SUMÁRIO DOS RESULTADOS OBTIDOS

	AUC			TPR@FPT, com Threshold t = 0.01		
	Ind.	**M1**	**M2**	**Ind,**	**M1**	**M2**
Kam	62.1	94.5	94.5	1.5	67.8	67.9
Far	93.5	92.2	92.9	19.9	66.5	66.8
Bec	91.5	92.1	92.2	54.2	66.3	66.7
Lok	91.4	93.8	93.7	78.0	72.7	73.1
Kit	74.6	94.5	94.3	18.9	71.7	71.1
Média	82.6	93.4	93.5	34.5	69.0	69.1

t=0.01 (1%) [17]. Para cada uma destas métricas foram testados os métodos: *(i)* Filtro Individual (**Ind.**): filtro bayesiano simples, sem recurso ao método distribuído elaborado neste trabalho; *(ii)* Método 1 (**M1**): Filtro que usa a média aritmética de todos os filtros para o cálculo da probabilidade final, e *(iii)* Método 2 (**M2**): Filtro que usa o Coeficiente de correlação de *Pearson* para o cálculo da probabilidade final.

A Tabela 1 descreve os resultados obtidos (valores médios de diversas simulações, em %) nos testes preliminares para cada uma das métricas e, dentro destas, para cada uma das técnicas utilizadas. Como é observado os métodos propostos obtêm um desempenho superior ao mecanismo

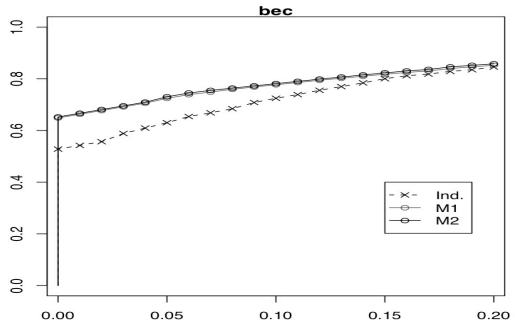

Fig. 4. Curva do ROC para o utilizador Bec

tradicionalmente utilizado. A Figura 4 representa a curva *ROC* para o utilizador *Bec*, utilizando os vários métodos acima descriminados (neste caso, ambos M1 e M2 têm performances similares). Através da análise de resultados pode-se concluir que tanto o métodos M1 como o M2 apresentam melhores resultados que o filtro individual na filtragem de mensagens *spam*. Embora sem confiança estatística, para o repositório estudado, o método M2 é ligeiramente melhor. De qualquer modo, ambos os métodos apresentam resultados encorajadores, quando comparados com o modelo individual. Em particular, existe uma diferença bastante significativa em termos da métrica TPR@FPR, o que é deveras relevante para a problemática específica da detecção de *spam*.

V. CONCLUSÕES E TRABALHO FUTURO

Nos dias correntes, a proliferação do *spam* acarreta problemas sérios pondo em causa a segurança e produtividade de utilizadores e empresas. Com a diminuição do *spam*, os níveis de segurança poderão aumentar e os custos normalmente associados a este fenómeno tenderão a diminuir.

A arquitectura aqui apresentada, e que se encontra já em fase de testes finais, propõe uma solução baseada em filtragem, sendo no entanto distribuída e com um cariz colaborativo, tirando pois partido da crescente colaboração e organização em redes sociais dos diversos intervenientes na rede Internet.

Neste contexto, este artigo descreveu em detalhe os diferentes componentes de uma arquitectura colaborativa para detecção de *spam*, tendo igualmente sido apresentados resultados iniciais da aplicação de solução proposta. Apesar de a partilha de filtros não levantar tantas questões de privacidade como no caso de se adoptar uma partilha de mensagens completas de correio electrónico, as questões de privacidade foram também consideradas neste projecto. É por esta razão que a solução proposta mantém o anonimato no nome dos ficheiros distribuídos e assenta num nó central autónomo e que se quer de confiança. No entanto, e apesar desta estratégia de armazenamento anónimo, como trabalho futuro pretende-se também adicionar ao sistema proposto capacidades de anonimato ao nível da rede. Desta forma, não será possível detectar a origem das conexões de rede das operações de transferência de filtros para o servidor, o que constitui um nível adicional de garantia de anonimato dos filtros submetidos pelos diversos intervenientes no sistema proposto.

AGRADECIMENTOS

Este trabalho é suportado pelo projecto FCT PTDC/EIA/64541/2006.

BIBLIOGRAFIA

[1] J. C. Sipior, B. T. Ward, and P. G. Bonner, "*Should Spam Be On The Menu?*," in Communications of the ACM, ACM, June 2004, pp. 59-64.

[2] Y. Zhou, M. S. Mulekar, and P. Nerellapalli, "*Adaptive Spam Filtering Using Dynamic Feature Space*," in 17th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'05, Nov. 2005.

[3] F. Fdez-Riverola, E. Iglesias, F. Diaz *et al.*, "*SpamHunting: An Instance-Based Reasoning System for Spam Labeling and Filtering*," in Decision Support Systems Journal, 2007, pp. 722-736.

[4] E. Blanzieri, and A. Bryl, "*A Survey of Learning-Based Techniques of Email Spam Filtering*," *In Technical Report # DIT-06-056*, Jan. 2008.

[5] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, "*Detecting Methods of Virus Email based on Mail Header and Encoding Anomaly*," *ICONIP*, Nov. 2008.

[6] A. Corporation, "*The Changing Face of Exchange E-mail Management*," Sept. 2005.

[7] R. Segal, "*Combining Global and Personal Anti-Spam Filtering*," in Proc. of the Fourth Conference on Email and Anti-Spam, Aug. 2007.

[8] A. C. Hubmann-Haidvogel, "*ThreadVis for Thunderbird: A Thread Visualisation Extension for the Mozilla Thunderbird Email Client*," Sept. 2008.

[9] "*Anti-Spam White Paper*," Aladdin Knowledge Systems.

[10] F. D. Garcia, J.-H. Hoepman, and J. v. Nieuwenhuizen, "*Spam Filter Analysis*," in Proceedings of 19th IFIP International Information Security Conference, WCC2004-SEC, Feb. 2004.

[11] A. Ramachandran, and N. Feamster, "*Understanding the Network-Level Behavior of Spammers*," in Proc. of SIGCOMM'06, Sept. 2006.

[12] P. Barford, and V. Yegneswaran, "*An Inside Look at Botnets*," Computer Sciences Department, University of Wisconsin, Madison.

[13] "*Whitepaper: Spam A Corporate Concern,*" *altoHiway Lim.,* Jan. 2004.

[14] S. Hird, "*Technical Solutions for Controlling Spam*," in Proc. of AUUG2002, Sept. 2002.

[15] TomFawcett, "*"Invivo"spam filtering: A challenge problem for data mining*," *KDD Explorations,* vol. 5, no. 2, pp. 140-148, Dec. 2003.

[16] R. Beckermann, A. McCallum, and G. Huang, "Automatic categorization of email into folders: benchmark experiments on Enron and SRI corpora," University of Massachusetts Amherst, 2004.

[17] M. Chang, W. Yih, and C. Meek, "Partitioned Logistic Re-gression for Spam Filtering," in 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2008, pp. 97–105.